Remarks

The above Amendments and these Remarks are in reply to the Office Action mailed March 3, 2008. Claims 1-65 are presented herewith for consideration. Applicants have amended claims 1-3, 6 -11, 13, 17, 19-20, 33-34, 38, 40, 48-49, 52, 58, and 62, and cancelled claims 16, 18, 32, 36-37 and 59.

I. Claim Objections

Claims 16 and 40 are objected to because of certain informalities. Applicants have amended claims 16 and 40 to correct the informalities objected to by the Examiner. Thus, Applicants request that the Examiner remove these claim objections.

II. Rejection Of Claims Under 35 U.S.C. §112

Claims 1-17, 38-58 and 60-65 were rejected under 35 U.S.C. §112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. Applicants have amended the rejected claims to meet the requirements set forth under 35 U.S.C. §112. Thus, Applicants request that the Examiner remove these rejections.

III. Rejection Of Claims Under 35 U.S.C. §102

Claims 20-23, 25, 27-31, 33 and 36 were rejected under 35 U.S.C. §102(e) as being anticipated by U.S. Patent No. 7,191,441 issued to Abbott et al. ("*Abbott*").

1. Independent Claim 20

Claim 20 recites:

> "compiling an application provided in a source language to create a first object code file;
>
> initializing the first object code file in a runtime environment to create a first application state; and

creating a serialized representation of the application objects in the runtime environment; and

building an optimized object code file using the serialized representation and the first object code file, wherein the optimized object code file includes instructions creating relation between objects in the runtime to create a second application state isomorphic to the first application state.

Abbott does not disclose "building an optimized object code file using the serialized representation and the first object code file, wherein the optimized object code file includes instructions creating relations between objects in the runtime to create a second application state isomorphic to the first application state." In contrast, *Abbott* discloses JIT-compiled code. JIT-compiled code is native machine code. *Abbott*, in col. 11, lines 26-36, discloses that:

> The save routine is able to handle both byte code and native code. If the method has not been compiled by JIT compiler 190 then the save only stores the Java byte code. On the other hand, if the method has been compiled, then the native (compiled) code will be stored as well.
>
> The JIT-compiled code for the method needs to be located and analysed to detect usage of absolute memory references. Relevant information from these addresses is then stored with the JIT-compiled code, so that the addresses can be properly adjusted when code is loaded back into memory.

The Examiner asserts that the JIT-compiled code is equivalent to the "optimized object code file" recited in claim 20. Applicants respectfully disagree. The "optimized object code file" is built "using the serialized representation and the first object code file." The "first object code file" represents a "first application state." If the byte code in *ABbott* is equivalent to the "first application state," the JIT-compiled native machine code is definitely not "a second application state isomorphic to the first application state." The term "isomorphism" is defined as "a one-to-one relation onto the map between two sets, which preserves the relations existing between elements in its domain." A JIT compiler does not preserve any one-to-one relation between byte code and native machine code. If the JIT-

compiled code disclosed in *Abbott* is the "first application state," *Abbott* does not disclose building another object code file using the JIT-compiled code. Therefore, the method recited in claim 20 is not anticipated by *Abbott*

Dependent claims 21-23, 25 and 27-31 depend directly or indirectly from independent claim 20. These dependent claims include all of the limitations of the independent claim from which they depend. Applicants respectfully assert that dependent claims 21-23, 25 and 27-31 are allowable for at least the reasons set forth above concerning independent claim 20.

2. Independent Claim 33

Claim 33 recites:

> "requesting an application from an application source server;
>
> receiving a first object code file loaded into a runtime environment and creating a first application state;
>
> receiving an optimized object code file using a serialized description of the application from the application source server and the first object code file, the optimized object code file including instructions creating relations between objects in the runtime; and
>
> loading the optimized object code file into a new runtime environment to create a second application state isomorphic to the first application state."

For at least the same reasons discussed above regarding claim 20, *Abbott* does not anticipate the method recited in claim 33. In particular, *Abbott* does not disclose "receiving an optimized object code file using a serialized description of the application from the application source server and the first object code file, the optimized object code file including instructions creating relations between objects in the runtime" or "loading the optimized object code file into a new runtime environment to create a second application state isomorphic to the first application state." As explained above, the JIT-compiled code

disclosed in *Abbott* is not equivalent to the "optimized object code file" recited in claim 20. Therefore, the method recited in claim 33 is also not anticipated by *Abbott*.

Dependent claim 36 depends directly or indirectly from independent claim 33. This dependent claim includes all of the limitations of the independent claim from which it depends. Applicants respectfully assert that dependent claim 36 is allowable for at least the reasons set forth above concerning independent claim 33.

III. Rejection Of Claims Under 35 U.S.C. §103

Claims 1-2, 4, 6-8, 11-17, 19, 38-39, 41-42, 44-45, 47-58 and 60-62 are rejected under 35 U.S.C. §103(a) as being unpatentable over *Abbott* in view of U.S. Patent No. 5,710,930 ("*Laney*").

Claims 3, 40 and 63-64 are rejected under 35 U.S.C. §103(a) as being unpatentable over *Abbott* in view of *Laney*, and further in view of U.S. Publication No. 2004/0044989 ("*Vachuska*").

Claims 5, 9-10, 43 and 46 are rejected under 35 U.S.C. §103(a) as being unpatentable over *Abbott* in view of *Laney*, in further view of U.S. Publication No. 2003/0195923 ("*Bloch*").

Claims 24, 26 and 35 are rejected under 35 U.S.C. §103(a) as being unpatentable over *Abbott* in view of *Bloch*.

Claim 34 is rejected under 35 U.S.C. §103(a) as being unpatentable over *Abbott* in view of U.S. Patent No. 5,987,256 ("*Wu*").

Claim 65 is rejected under 35 U.S.C. §103(a) as being unpatentable over *Abbott* in view of *Laney* and *Vachuska*, and further in view of *Bloch*.

Abbott* in view of *Laney

1. Independent Claim 1

Claim 1 recites:

> "creating a serialized representation of application objects in the runtime environment;
>
> building an optimized object code file using the serialized representation and the first object code file, wherein the optimized object code file includes instructions creating relations between objects in the runtime environment; and
>
> loading the optimized object code file into a new runtime environment to create a second application state isomorphic to the first application state."

The Examiner sets forth that it would have been obvious to one of ordinary skill in the art to incorporate the teaching of *Laney* into the teaching of *Abbott* to disclose the method recited in claim 1. Applicants respectfully disagree. Neither *Abbott* nor *Laney*, alone or in combination, disclose "building an optimized object code file using the serialized representation and the first object code file" and "loading the optimized object code file into a new runtime environment to create a second application state isomorphic to the first application state."

In contrast to the method recited in claim 1, both *Abbott* and *Laney* require a special "loader program" to restore the application. *Abbott* discloses that

> the saved state [of the application] is loaded by a special 'javaload' tool, which loads the stored application into a Java VM, bypassing all the initialization normally associated with bringing up a Java VM. The reloading of the application is performed in stages. [emphasis added] Col. 18, lines 10-13.

Fig. 8 of *Abbott* illustrates the steps executed by the "special 'javaload' tool" to restore the application. In contrast, the method recited in claim 1 builds an "optimized object

code using the serialized representation and the first object code file." The method recited in claim 1 does not require a special "loader tool" to restore the application. The "optimized object code" is simply executed by the "new runtime environment."

In addition, if the Examiner equates the JIT-compiled code to the "optimized object code" recited in claim 1, *Abbott* does not disclose "loading the optimized object code file into a new runtime environment to create a second application state isomorphic to the first application state." The "first application state" recited in claim 1 is associated with a "first object code file." The byte code in *Abbott* cannot constitute an "object code file." Thus, even if the JIT-compiled code in *Abbott* were equivalent to the "optimized object code" recited in claim 1, the JIT-compiled code, when loaded into a runtime environment would not create a second application state isomorphic to the first application state." The term "isomorphism" is defined as "a one-to-one relation onto the map between two sets, which preserves the relations existing between elements in its domain." A JIT-compiler does not preserve any one-to-one relation between byte code and native machine code. Therefore, the method recited in claim 1 is not obvious over *Abbott*.

Laney does not provide the elements missing in *Abbott*. *Laney* also requires a special "loader program" to restore the application. *Laney*, in col. 6, line 64 – col. 7, line 2, discloses that:

> The loader program (i.e., initialization code) stored in boot sector 46 of mass storage device 26 is replaced with a new loader program. The new loader program is created by the shutdown program of the present invention that contains the address to the shutdown program's 'restart' entry point. This means that the new loader program contains the "restart" entry address of the shutdown program and will branch to the shutdown program at system power up.

As shown above, *Laney* relies on creating a new loader program that will be executed upon restarting the application. *Laney*, in col. 7, lines 19-26, discloses the restart process of the shutdown program using the new loader program:

the BIOS loader program loads the special loader program from boot sector 46 of mass storage device 26 into system memory 23. At step 62, the special loader program branches to the 'restart' entry point of the shutdown program. At step 63, the shutdown program accesses shutdown state buffer 43 in system memory 23 to restore the system state of computer system 20. [emphasis added].

As discussed above, both *Abbott* and *Laney* require the use of a separate "loader program" to restore the application. The method recited in claim 1 eliminates the need for a special "loader program" to create a "second application state isomorphic to the first application state." The "second application state" recited in claim 1 is created by executing the "optimized object code" which already includes "relations between objects in the runtime environment." In executing the "optimized object code," all values are already calculated and are simply assigned by the new runtime environment. Therefore, the method recited in claim 1 is not obvious over *Abbott* in view of *Laney*.

Dependent claims 2, 4, 6-8, 11-17 and 19 depend directly or indirectly from independent claim 1. These dependent claims include all of the limitations of the independent claim from which they depend. Applicants respectfully assert that dependent claims 2, 4, 6-8, 11-17 and 19 are allowable for at least the reasons set forth above concerning independent claim 1.

2. Independent Claim 38

Claim 38 recites:

"creating a serialized representation of application objects in the runtime environment;

building an optimized object code file using the serialized representation of the application objects and the first object code file, wherein the optimized object code file includes instructions creating relations between objects in the runtime environment; and

loading the optimized object code file into a new runtime environment to create a second application state isomorphic to the first application state."

For at least the same reasons discussed above regarding claim 1, the method recited in claim 38 is not obvious over *Abbott* in view of *Laney*. In particular, the combination of *Abbott* and *Laney* do not disclose "building an optimized object code file using the serialized representation of the application objects and the first object code file, wherein the optimized object code file includes instructions creating relations between objects in the runtime environment" or "loading the optimized object code file into a new runtime environment to create a second application state isomorphic to the first application state." Therefore, the method recited in claim 38 is not obvious over the combination of *Abbott* and *Laney*.

Dependent claims 39, 41, 42, 44-45 and 47-51 depend directly or indirectly from independent claim 38. These dependent claims include all of the limitations of the independent claim from which they depend. Applicants respectfully assert that dependent claims 39, 41, 42, 44-45 and 47-51 are allowable for at least the reasons set forth above concerning independent claim 38.

3. Independent Claim 52

Claim 52 recites:

"compiling the application into a first object code file;

loading the first object code file into a first runtime environment to create a first application state;

creating a serialized representation of a memory space in said first runtime environment;

building a second object code file using said serialized representation and the first object code file, wherein the second object code file includes

instructions creating relations between objects in the runtime environment; and

loading said second object code into a second runtime environment to create a second application state isomorphic to the first application state."

For at least the same reasons discussed above regarding claim 1, the method recited in claim 52 is not obvious over *Abbott* in view of *Laney*. In particular, the combination of *Abbott* and *Laney* do not disclose "building a second object code file using said serialized representation and the first object code file, wherein the second object code file includes instructions creating relations between objects in the runtime environment" or "loading said second object code into a second runtime environment to create a second application state isomorphic to the first application state." Therefore, the method recited in claim 52 is not obvious over the combination of *Abbott* and *Laney*.

Dependent claims 53-57 depend directly or indirectly from independent claim 52. These dependent claims include all of the limitations of the independent claim from which they depend. Applicants respectfully assert that dependent claims 53-57 are allowable for at least the reasons set forth above concerning independent claim 52.

4. Independent Claim 58

Claim 58 recites:

"receiving from a runtime environment a serialized representation of objects in a memory space of the runtime environment; and

building an optimized object code file using the serialized representation and a compiled object code file used to create the memory space, wherein the optimized object code file includes instructions creating relations between objects in the runtime environment."

For at least the same reasons discussed above regarding claim 1, the method recited in claim 58 is not obvious over *Abbott* in view of *Laney*. In particular, the combination of *Abbott* and *Laney* do not disclose "building an optimized object code file using the serialized representation and a compiled object code file used to create the memory space, wherein the optimized object code file includes instructions creating relations between objects in the runtime environment." Therefore, the method recited in claim 58 is not obvious over the combination of *Abbott* and *Laney*.

Dependent claims 59-61 depend directly or indirectly from independent claim 58. These dependent claims include all of the limitations of the independent claim from which they depend. Applicants respectfully assert that dependent claims 59-61 are allowable for at least the reasons set forth above concerning independent claim 58.

5. Independent Claim 62

Claim 62 recites:

> "creating a serialized representation of application objects in the runtime environment;
>
> building an optimized object code file using the serialized representation and the first object code file, wherein the optimized object code file includes instructions creating relations between objects in the runtime environment; and
>
> loading the optimized object code file into a new runtime environment via the network to create a second state isomorphic to the first application state."

For at least the same reasons discussed above regarding claim 1, the method recited in claim 62 is not obvious over *Abbott* in view of *Laney*. In particular, the combination of *Abbott* and *Laney* does not disclose "building an optimized object code file using the serialized representation and the first object code file, wherein the optimized object code file

includes instructions creating relations between objects in the runtime environment" or "loading the optimized object code file into a new runtime environment via the network to create a second state isomorphic to the first application state." Therefore, the method recited in claim 62 is not obvious over the combination of *Abbott* and *Laney*.

Abbott* in view of *Laney*, and further in view of *Vachuska

1. Dependent Claim 3

Claim 3 depends from independent claim 1. Claim 1, in part, recites "building an optimized object code file using the serialized representation and the first object code file [that] includes instructions creating relations between objects in the runtime environment" that is subsequently loaded into a new run time environment. As discussed above, the combination of *Abbott* and *Laney* do not disclose this claim element. *Vachuscka* does not provide the elements missing from *Abbott*. The technology disclosed in *Vachuska* cannot be incorporated into *Abbott-Laney* to disclose "optimized object code file [that] includes instructions creating relations between objects in the runtime environment." *Vachuska* discloses a source-code generator which uses a reflection mechanism to analyze source templates. Nothing disclosed in *Vachuska* can be incorporated into *Abbott* to disclose the method recited in claim 1. Thus, Applicants respectfully suggest that the method recited in claim 1 is not obvious over *Abbott* in view of *Laney*, in further view of *Vachuska*. Because claim 3 depends from claim 1, claim 3 is also not obvious over *Abbott* in view of *Laney*, in further view of *Vachuska*.

2. Dependent Claim 40

Claim 40 depends from independent claim 38. Claim 38, in part, recites "building an optimized object code file using the serialized representation of the application objects and the first object code file [that] includes instructions creating relations between objects in the runtime environment." For at least the same reasons discussed above regarding claims 3 and 38, Applicants respectfully suggest that the method recited in claim 38 is not obvious over

Abbott in view of *Laney*, in further view of *Vachuska*. Because claim 40 depends from claim 38, claim 40 is also not obvious over *Abbott* in view of *Laney*, in further view of *Vachuska*.

3. Dependent Claims 63 and 64

Claims 63 and 64 depend from independent claim 62. Claim 62, in part, recites "building an optimized object code file using the serialized representation and the first object code file [that] includes instructions creating relations between objects in the runtime environment." For at least the same reasons discussed above regarding claims 3 and 62, Applicants respectfully suggest that the method recited in claim 62 is not obvious over *Abbott* in view of *Laney*, and further in view of *Vachuska*. Because claims 63 and 64 depend from claim 62, claims 63 and 64 are also not obvious over *Abbott* in view of *Laney*, and further in view of *Vachuska*.

Abbott* in view of *Laney*, in further view of *Bloch

1. **Dependent Claims 5 and 9-10**

Claims 5 and 9-10 depend directly or indirectly from independent claim 1. Claim 1, in part, recites "building an optimized object code file using the serialized representation and the first object code file [that] includes instructions creating relations between objects in the runtime environment" that is subsequently loaded into a new run time environment. As discussed above with regard to claim 1, the combination of *Abbott* and *Laney* does not disclose this claim element.

Bloch does not provide the elements missing from the combination of *Abbott* and *Laney*. *Bloch* discloses a presentation server "designed to receive a mark-up language description of a user-interface and dynamically compile that mark-up language description to executable code." ¶[0030]. The presentation server disclosed in *Bloch* does not perform any of the steps recited in claim 1, which recites "a method for decreasing a computer application start-up time." Therefore, the method recited in claim 1 is not obvious over the combination

of *Abbott*, *Laney* and *Bloch*. Because claims 5 and 9-10 depend from claim 1, claims 5 and 9-10 are also not obvious over the combination of *Abbott*, *Laney* and *Bloch*.

2. **Dependent Claim 43 and 46**

Claims 43 and 46 depend from independent claim 38. Claim 38, in part recites "building an optimized object code file using the serialized representation of the application objects and the first object code file [that] includes instructions creating relations between objects in the runtime environment." For at least the same reasons discussed above regarding claims 5 and 9-10, the method recited in claim 38 is not obvious over the combination of *Abbott*, *Laney* and *Bloch*. Because claims 43 and 46 depend from claim 38, claims 43 and 46 are also not obvious over *Abbott* in view of *Laney*, in further view of *Bloch*.

Abbott* in view of *Bloch

Dependent Claims 24 and 26 depend from independent claim 20. Claim 20, in part, recites "building an optimized object code file using the serialized representation and the first object code file, wherein the optimized object code file includes instructions creating relation between objects in the runtime to create a second application state isomorphic to the first application state." For at least the reasons discussed above regarding claim 20, *Abbott* does not disclose this claim element.

Bloch does not disclose the elements missing from *Abbott*. *Bloch* discloses a presentation server "designed to receive a mark-up language description of a user-interface and dynamically compile that mark-up language description to executable code." ¶[0030]. The presentation server disclosed in *Bloch* does not perform this steps recited in claim 20. Therefore, the method recited in claim 20 is not obvious over *Abbott* in view of *Bloch*. Because claims 24 and 26 depend from claim 20, claims 24 and 26 are also not obvious over *Abbott* in view of *Bloch*.

Dependent claim 35 depends from independent claim 33. Claim 33, in part, recites, "receiving an optimized object code file using a serialized description of the application from

the application source server and the first object code file, the optimized object code file including instructions creating relations between objects in the runtime" and "loading the optimized object code file into a new runtime environment to create a second application state isomorphic to the first application state." For at least the same reasons discussed above regarding claim 20, the method recited in claim 33 is not obvious over *Abbott* in view of *Bloch*. Because claim 35 depends from claim 33, the method recited in claim 35 is also not obvious over *Abbott* in view of *Bloch*.

Abbott* in view of *Wu

Claim 34 depends from independent claim 33. Claim 33, in part, recites receiving object code and subsequently "loading the object code received from the application source server into a runtime environment." The object code includes "instructions for replicating a runtime memory state of the application." As described above with regard to claim 35, *Abbott* does not create object code replicating the runtime memory state of the application and load the object code into a new runtime environment. In contrast, *Abbott* saves a "snapshot" of the application in the saved state and rebuilds the application component by component when a restore routine is called. Therefore, Applicants respectfully suggest that the method recited in claim 33 is not obvious over *Abbott*.

Moreover, *Wu* does not disclose the elements missing from *Abbott*. The Examiner cites *Wu* to teach that object code may include media assets. Applicants respectfully suggest that that technology disclosed in *Wu* cannot be incorporated into *Abbott* to teach the method recited in claim 33. Incorporating object code that includes media assets into *Abbott* does not disclose receiving object code and subsequently "loading the object code received from the application source server into a runtime environment." Therefore, Applicants respectfully suggest that the method recited in claim 33 is not obvious over *Abbott* in view of *Wu*. Because claim 34 depends from claim 33, claim 34 is also not obvious over *Abbott* in view of *Wu*.

Abbott* in view of *Laney* and *Vachuska*, and further in view of *Bloch

Dependent Claim 65 depends from independent claim 62. Claim 62, in part, recites "building an optimized object code file using the serialized representation and the first object code file [that] includes instructions creating relations between objects in the runtime environment." For at least the reasons discussed above regarding claims 63 and 64, the combination of *Abbott*, *Laney* and *Vachuska* does not disclose this claim element.

Bloch does not disclose the elements missing from the combination of *Abbott*, *Laney* and *Vachuska*. *Bloch* discloses a presentation server "designed to receive a mark-up language description of a user-interface and dynamically compile that mark-up language description to executable code." ¶[0030]. The presentation server disclosed in *Bloch* does not perform this "building an optimized code" step recited in claim 62. Therefore, the method recited in claim 62 is not obvious over the combination of *Abbott*, *Laney*, *Vachuska* and *Bloch*. Because claim 65 depends from claim 62, claim 65 is also not obvious over combination of *Abbott*, *Laney*, *Vachuska* and *Bloch*.

Additional Remarks

Based on the above amendments and these remarks, reconsideration of claims 1-15, 17, 19-31, 33-35, 38-58 and 60-65 is respectfully requested.

The Examiner's prompt attention to this matter is greatly appreciated. Should further questions remain, the Examiner is invited to contact the undersigned attorney by telephone.

The Commissioner is authorized to charge any underpayment or credit any overpayment to Deposit Account No. 501826 for any matter in connection with this response, including any fee for extension of time, which may be required.

Respectfully submitted,

Date: August 1, 2008

By: */Scott D. Sanford/*
Scott D. Sanford
Reg. No. 51,170

VIERRA MAGEN MARCUS & DENIRO LLP
575 Market Street, Suite 2500
San Francisco, California 94105
Telephone: (415) 369-9660
Facsimile: (415) 369-9665

ssanford@vierramagen.com